University of Waterloo

CLICKBAIT DETECTION CHALLENGE SPOILER CLASSIFICATION AND GENERATION USING TRANSFORMER BASED MODELS

Raavi Deveswara Sai Praveen
21007013

Jay Nidumolu
21108165

Abstract

The Clickbait Spoiler Generation competition challenges participants to develop models capable of generating spoilers for clickbait posts, aiming to enhance reader satisfaction by providing core content upfront. Clickbait headlines, often exaggerated or misleading, entice readers to click through but frequently leave them disappointed with the content. By generating spoilers that reveal essential information of the clickbait post, we aim to mitigate this issue. This report documents our comprehensive approach to tackling this challenge, detailing the methodology, experiments, and results achieved throughout the competition.

We addressed two primary tasks in this competition. For Task 1, which focused on spoiler classification, we employed a diverse set of pre-trained transformer models, including BART, RoBERTa, and DeBERTa. These models were fine-tuned on the dataset to accurately classify spoiler types, leveraging their contextual understanding and language modeling capabilities.

For Task 2, which involved spoiler generation, we explored several transformer-based models, such as BART-base, BART-large-CNN, and T5. These models were fine-tuned to adapt to the specific task of generating high-quality spoilers that closely match the reference spoilers.

In summary, our approach utilized a selection of pre-trained transformer models to address the challenge of clickbait spoiler generation effectively.

1 Introduction

Clickbait spoiling aims at generating short texts that satisfy the curiosity induced by click- bait posts generally posted on Reddit, Twitter, and Facebook. This project is a derivative of the Clickbait Challenge [5] at SemEval 2023 and aims to solve two subtasks from the challenge i.e. spoiler type classification (phrase, passage,multi-line) followed by spoiler generation.

Clickbaits often don't contain enough relevant information and meant to have attractive titles to draw user's attention. They often are used as source to show advertisement or convey something obvious. Generally, clickbait spoilers can be of 3 types: phrase, passage, and multi-line based. As seen from example shown in Figure 1, all 3 spoiler types have different text structure, length etc. which demands that we have different methods for generating them.

Figure 1: Example of Twitter Clickbaits

This project is divided into two distinct subtasks. The first sub-task involves classifying generated spoilers into one of three predefined categories. The second sub-task focuses on generating the actual spoilers for given texts. The dataset for this competition is structured into three JSON files: one for training, one for validation, and one for testing. Both the training and validation datasets include columns such as post text, target title, target paragraphs, along with the spoiler and its type. Conversely, the test dataset contains

all these columns except for the spoilers and their types. These JSON files were preprocessed and converted into CSV format, merging the original columns into a single column for simplicity.

In our approach, we experimented with fine-tuning various transformer-based models for both spoiler classification and generation tasks. Notably, this competition differs from others on Kaggle in that it did not provide any starter code. The code for the task is provided in the github [1] For evaluation, we utilized the F1-score to measure performance on the classification task (sub-task 1) and the METEOR score to assess the quality of generated spoilers (sub-task 2). We compared the results of all models using these metrics on both the validation and test datasets to determine their effectiveness.

2 Related Work:

2.1 Transformer Based Models:

Transformers have revolutionized natural language processing (NLP) with their ability to handle various tasks effectively. Several transformer-based models have significantly advanced the state-of-the-art in tasks such as text classification, generation, and understanding. In this section, we provide an overview of some key transformer-based models used in our work.

2.1.1 *BERT (Bidirectional Encoder Representations from Transformers)*:

BERT, introduced by Devlin et al. (2018) [4], represents a groundbreaking advancement in NLP due to its bidirectional attention mechanism. Unlike traditional models that read text sequentially, BERT reads the entire sequence of words simultaneously. This bidirectional approach enables the model to understand context from both the left and right of each word, leading to a more nuanced understanding of language. BERT's ability to capture the intricate relationships between words makes it highly effective for tasks such as question answering, sentiment analysis, and named entity recognition.

The training process for BERT involves two main steps: pre-training and fine-tuning. During pre-training, BERT uses a masked language model (MLM) objective, where random words in a sentence are masked, and the model is trained to predict these missing words. Additionally, BERT employs the Next Sentence Prediction (NSP) task to understand the relationship between sentence pairs. These pre-training tasks allow BERT to build a deep understanding of language patterns, which can be fine-tuned for specific tasks by adding a task-specific output layer.

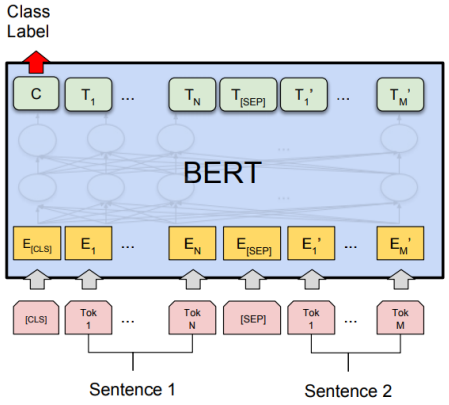

Figure 2: Archtechture of BERT

BERT has set new benchmarks in various NLP tasks, including the Stanford Question Answering Dataset (SQuAD) and the General Language Understanding Evaluation (GLUE) benchmark. Its impact extends beyond these benchmarks, influencing subsequent models and methodologies in the field. The success of BERT has led to the development of numerous variants and extensions, making it a cornerstone of modern NLP research.

2.1.2 *RoBERTa (Robustly operated BERT approach)*:

RoBERTa, introduced by Liu et al. (2019) [12], is an optimized version of BERT that enhances its performance by refining the training process. One of the key differences between RoBERTa and BERT is the removal of the Next Sentence Prediction (NSP) objective. RoBERTa focuses solely on the masked language modeling (MLM) task, which has been shown to improve performance on various NLP benchmarks.

RoBERTa's training process involves using a larger corpus and longer training times compared to BERT. The model is trained on more extensive data and uses larger batch sizes, which contribute to its robustness and performance. Additionally, RoBERTa employs dynamic masking, where the masked words are randomly chosen during each training epoch, providing a more varied learning experience.

[1] https://github.com/praveen-rds/MSCI-641-Final-Project

The improvements introduced by RoBERTa have resulted in significant gains in performance across several benchmarks, including the GLUE benchmark and the Stanford Question Answering Dataset (SQuAD). RoBERTa's enhanced training regimen has made it a popular choice for researchers and practitioners seeking high-performance language models for a range of NLP tasks.

2.1.3 *DeBERTa (Decoding-enhanced BERT with Disentangled Attention)*:

DeBERTa, developed by He et al. (2021) [7], builds upon the BERT architecture by introducing disentangled attention mechanisms and enhanced decoding strategies. The model's key innovation is the disentangled attention mechanism, which separates the modeling of content and position information. This approach allows DeBERTa to better capture the dependencies between words and their contextual meanings.

The disentangled attention mechanism in DeBERTa enables the model to more effectively handle complex language patterns and relationships. Additionally, DeBERTa uses enhanced decoding strategies to improve the generation of text and the understanding of language. These advancements contribute to the model's superior performance on various NLP tasks, including text classification and generation.

DeBERTa has achieved state-of-the-art results on several benchmarks, such as the GLUE benchmark and the SQuAD dataset. Its ability to capture nuanced language features and handle diverse tasks has made it a valuable addition to the toolkit of modern NLP models. The innovations introduced by DeBERTa have influenced subsequent research and development in the field.

2.1.4 *BART (Bidirectional Auto-Regressive Transformer):*

BART, proposed by Lewis et al. (2020) [11], combines the strengths of bidirectional and autoregressive models. This hybrid approach leverages a bidirectional encoder to understand the context and an autoregressive decoder to generate text. BART's architecture is particularly suited for sequence-to-sequence tasks, such as text summarization and translation, where both understanding and generating text are crucial.

The model's pre-training involves a denoising autoencoder objective, where various forms of noise are introduced to the text, such as token masking, sentence permutation, and text deletion. The model learns to reconstruct the original text from these corrupted versions. This training strategy allows BART to handle a wide range of text generation tasks effectively, as it learns to deal with incomplete or noisy input data.

BART has demonstrated impressive performance across various NLP tasks, including summarization, translation, and question answering. Its ability to generate coherent and contextually appropriate text has made it a valuable tool for applications that require high-quality text generation. BART's versatility and effectiveness have led to its adoption in both academic research and practical applications.

2.1.5 *T5 (Text-To-Text Transfer Transformer)*:

T5, proposed by Raffel et al. (2020) [16], is a versatile model that treats every NLP task as a text-to-text problem. By framing all tasks as generating text from input text, T5 achieves a high degree of flexibility and generalization. This unified approach allows the model to handle a wide range of tasks, from text classification to translation, using a consistent framework.

The training of T5 involves a text-to-text pre-training objective, where the model learns to generate text based on various input transformations. This training strategy enables T5 to effectively handle different types of text generation and understanding tasks. The model's ability to generalize across tasks makes it a powerful tool for applications requiring diverse NLP capabilities.

T5 has demonstrated exceptional performance on multiple benchmarks, including the GLUE benchmark and the SuperGLUE benchmark. Its unified framework and strong performance across a wide range of tasks have made it a prominent model in the field of NLP. The approach taken by T5 has influenced the development of future models and methodologies in text generation and understanding.

3 Approach:

Our project focuses on Clickbait detection, which is divided into two tasks: Task 1 involves identifying the spoiler type for each clickbait article, a classification problem, and Task 2 involves generating spoilers, a text generation problem.

Task 1: Spoiler Type Classification For Task 1, we explored various classical machine learning models and deep neural networks. However, our best-performing models were BART, RoBERTa, and DeBERTa. We fine-tuned the hyper-parameters of these pre-existing models to achieve optimal results. By leveraging the robust capabilities of these transformer models, we successfully surpassed the baseline implementation scores. We used the F1 score as our primary evaluation metric, which helped us understand which target class was classified erroneously by each model.

Task 2: Spoiler Generation For Task 2, we employed BART-base, BART-large-CNN, and T5 models due to their strong text generation capabilities. These models were fine-tuned to produce relevant and coherent spoilers for clickbait articles. The generated spoilers were evaluated using the Meteor score, which measures the quality of the text generation by comparing it to human-written references.

By combining these approaches, we aimed to address both the classification and generation aspects of clickbait detection, ensuring comprehensive coverage of the problem space.

3.1 Data Pre Processing

Before feeding the data into any model, several common preprocessing steps were applied to ensure that the dataset was clean and well-structured. We loaded the dataset from JSONL files and converted them into pandas DataFrames for further processing. Text data underwent preprocessing to standardize and clean it. This involved converting text to lowercase and removing punctuation and non-alphanumeric characters.To create a unified text representation, different text features from each entry were integrated. This involved concatenating 'postText', 'targetParagraphs', and 'targetTitle' into a single list of text elements for each entry. The integrated text features and labels were saved into CSV files for training and validation datasets. The test dataset was prepared similarly, but without labels.
These preprocessing steps ensure that the text data is clean, consistent, and well-structured, providing a solid foundation for model training and evaluation. For every work from this point, the same csv files that were generated in this step have been utilized.

3.2 Preprocessing for Neural Networks and Word Embeddings

The preprocessing pipeline involves several key steps to prepare the data for training and evaluation on individual models. We load our datasets using pandas and convert the text and labels into lists for further processing. We used 'LabelEncoder' to encode categorical labels into numeric values. This step ensures that the labels are in a format suitable for training our model. The tokenizer handles truncation, padding, and encoding of text sequences. Each sequence is padded to a maximum length of 512 tokens, ensuring uniform input size for the neural network. We create a custom dataset class to manage our encoded data and labels, making it compatible with PyTorch's DataLoader. These preprocessing steps ensure that our data is properly formatted and prepared for training the neural network models.

For our clickbait detection project, we employed a range of models with their respective tokenizers to handle both classification and text generation tasks. For the classification of spoiler types, we experimented with several models, each with its own tokenizer which includes BART, roBERTa and deBERTa. Each tokenizer was used to convert text inputs into token IDs compatible with the corresponding model, enabling effective handling of semantic information for classification. Similarly, for the text generation task, we used BART and T5 models with their respective tokenizers.The use of different tokenizers allowed us to leverage the strengths of each model architecture for both classification and generation tasks, ensuring comprehensive coverage.

3.3 Transformer Models

In our project, we utilized several transformer-based models to address the tasks of clickbait detection and spoiler generation. These models include BART, RoBERTa, and DeBERTa, for task 1 and BART, and T5 for task 2, each known for their robust performance in natural language processing tasks. Here, we provide a detailed overview of the models and their configurations.

3.3.1 *BartForSequenceClassification*

The `BartForSequenceClassification` was used for our classification task. BART (Bidirectional and Auto-Regressive Transformers) is a sequence-to-sequence model that combines the bidirectional context of BERT with the autoregressive nature of GPT. It has proven effective in various text generation and classification tasks.

Model Architecture:

- **Embedding Layer:** The model uses a shared embedding layer for both the encoder and decoder, with an embedding size of 768.
- **Encoder:** The encoder consists of 6 layers, each with multi-headed self-attention mechanisms and feed-forward neural networks.
- **Decoder:** The decoder also has 6 layers, equipped with both self-attention and encoder-decoder attention mechanisms.
- **Classification Head:** The classification head includes a dense layer with a dropout and an output projection layer with 3 classes (as per our classification task).

Training Configuration:

- **Tokenizer:** We used `BartTokenizer` for tokenization.
- **Max Sequence Length:** 512 tokens.
- **Learning Rate:** Fine-tuned as required for optimal performance, which was achieved at 5e-5.

3.3.2 *RoBERTa*

For the classification task, we also employed the `RobertaForSequenceClassification`. RoBERTa (A Robustly Optimized BERT Pretraining Approach) enhances the BERT model by optimizing training techniques and using more data.

Model Architecture:

- **Embedding Layer:** RoBERTa uses a word embeddings layer with an embedding size of 768.
- **Encoder:** The encoder has 12 layers, each with multi-headed self-attention and feed-forward neural networks.
- **Classification Head:** The classification head comprises a dense layer, dropout, and output projection layer with 3 classes.

Training Configuration:

- **Tokenizer:** We used `RobertaTokenizer` for tokenization.
- **Max Sequence Length:** 512 tokens.
- **Dropout:** Various dropout rates were experimented with, primarily 0.2 for regularization.
- **Learning Rate:** Fine-tuned across different runs for best results, best results were given with a rate of 3e-5 and a weight decay of 0.01.

3.3.3 *DeBERTa*

DeBERTa (Decoding-enhanced BERT with Disentangled Attention) provided the best performance in our classification task. DeBERTa improves on BERT by introducing disentangled attention and an enhanced mask decoder.

Model Architecture:

- **Embedding Layer:** DeBERTa employs disentangled attention mechanisms in its embedding layers.
- **Encoder:** The encoder comprises several layers, with disentangled attention heads that separate content and position information.
- **Classification Head:** Includes a dense layer and an output layer suitable for our classification task.

Training Configuration:

- **Tokenizer:** We used the tokenizer associated with DeBERTa for tokenization.
- **Max Sequence Length:** 512 tokens.
- **Optimizer:** AdamW optimizer with a learning rate of 3e-5 and weight decay of 0.01.
- **Scheduler:** A linear learning rate scheduler with warm-up was used, with 10% of the total training steps allocated for warm-up.
- **Training Duration:** 10 epochs.

3.3.4 *BartForConditionalGeneration (BART-base)*

The `BartForConditionalGeneration` was used for our text generation tasks. BART (Bidirectional and Auto-Regressive Transformers) is a sequence-to-sequence model that combines the bidirectional context of BERT with the autoregressive nature of GPT, making it effective for text generation.

Model Architecture:

- **Embedding Layer:** The model uses a shared embedding layer for both the encoder and decoder, with an embedding size of 768.
- **Encoder:** The encoder consists of 12 layers, each with multi-headed self-attention mechanisms and feed-forward neural networks.
- **Decoder:** The decoder also has 12 layers, equipped with both self-attention and encoder-decoder attention mechanisms.
- **Generation Head:** The generation head includes a dense layer with a dropout and an output projection layer for token prediction.

Training Configuration:

- **Tokenizer:** We used `BartTokenizer` for tokenization.
- **Max Sequence Length:** 512 tokens.
- **Learning Rate:** Fine-tuned as required for optimal performance, which was achieved at 5e-5.
- **Learning Rate Scheduler:** A linear learning rate scheduler was employed to adjust the learning rate during training, improving convergence.
- **Batch Size:** 8

3.3.5 *BartForConditionalGeneration (BART-large-CNN)*

The `BartForConditionalGeneration` with the BART-large-CNN configuration was also used for text generation. This variant is fine-tuned for summarization tasks and features enhanced capabilities for generating concise and coherent text.

Model Architecture:

- **Embedding Layer:** The model uses a shared embedding layer for both the encoder and decoder, with an embedding size of 1024.
- **Encoder:** The encoder consists of 12 layers, each with multi-headed self-attention mechanisms and feed-forward neural networks.
- **Decoder:** The decoder also has 12 layers, equipped with both self-attention and encoder-decoder attention mechanisms.
- **Generation Head:** The generation head includes a dense layer with a dropout and an output projection layer for token prediction.

Training Configuration:

- **Tokenizer:** We used `BartTokenizer` for tokenization.
- **Max Sequence Length:** 512 tokens.
- **Learning Rate:** Fine-tuned as required for optimal performance, which was achieved at 5e-5.
- **Batch Size:** 4

3.3.6 *T5ForConditionalGeneration (T5-base)*

The `T5ForConditionalGeneration` was employed for our text generation tasks. T5 (Text-To-Text Transfer Transformer) is a versatile transformer model designed to convert every NLP problem into a text-to-text format, allowing a unified approach to various tasks.

Model Architecture:

- **Embedding Layer:** The model uses a shared embedding layer for both the encoder and decoder, with an embedding size of 768.
- **Encoder:** The encoder consists of 12 layers, each with multi-headed self-attention mechanisms and feed-forward neural networks.
- **Decoder:** The decoder also has 12 layers, equipped with both self-attention and encoder-decoder attention mechanisms.
- **Generation Head:** The generation head includes a dense layer with a dropout and an output projection layer for token prediction.

Training Configuration:

- **Tokenizer:** We used `T5Tokenizer` for tokenization.
- **Max Sequence Length:** 512 tokens.
- **Learning Rate:** Fine-tuned as required for optimal performance, which was achieved at 5e-5.
- **Batch Size:** 8.
- **Learning Rate Scheduler:** A learning rate scheduler was employed to adjust the learning rate during training, improving convergence.

Each of these models was fine-tuned on our dataset, with specific configurations tailored to maximize performance in our classification and generation tasks. We utilized pretrained weights and tokenizers specific to each model, ensuring robust embedding representations and efficient training processes.

4 Experiments:

4.1 Datasets:

The dataset contains the clickbait posts and manually cleaned versions of the linked documents, and extracted spoilers for each clickbait post (the dataset was constructed and published in the corresponding paper). Additionally, the spoilers are categorized into three types: short phrase spoilers, longer passage spoilers, and multiple non-consecutive pieces of text.

The training, validation, and test data are available for download in the "Data" tab. This dataset contains 3,200 posts for training (the file training.jsonl) and 400 posts for validation (the file val.jsonl). After training and validation, systems are evaluated on 400 test posts.

In the mentioned 3200 posts from the train data, 1367 posts have the Phrase type spoiler class which is 42.7%, 1274 posts have Passage type spoiler class which is 39.8% and remaining are of multi class spoiler type.

4.2 Training:

Training was conducted using the models detailed in the "Approach" section, specifically BART, RoBERTa, DeBERTa, and T5. For Task 1, which involves classification, BART, RoBERTa, and DeBERTa were employed to categorize spoilers into predefined types. For Task 2, which focuses on generating spoilers, BART and T5 were utilized. The specific training configurations, including batch sizes, learning rates, epochs, and the use of learning rate schedulers, are outlined in the "Approach" section. These configurations were selected to ensure effective training of the models for both classification and generation tasks, balancing performance and training efficiency.

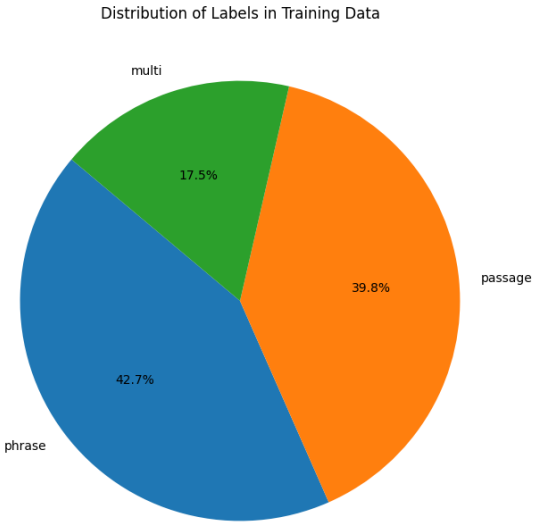

Figure 3: Spoiler Class Distribution in Train Data.

4.3 Evaluation Metrics:

The evaluation metrics used for this Clickbait Spoiler Detection are F1 score and METEOR score for the respective tasks.

4.3.1 *F_1-Score:*

The **$\mathbf{F_1}$ score** measures accuracy using the statistics precision p and recall r. Precision is the ratio of true positives t_p to all predicted positives $(t_p + f_p)$. Recall is the ratio of true positives t_p to all actual positives $(t_p + f_n)$. The F1 score is given by:

$$F_1 = 2\frac{p \cdot r}{p + r}$$

where

$$p = \frac{t_p}{t_p + f_p}, \quad r = \frac{t_p}{t_p + f_n}$$

The F_1 metric weights recall and precision equally, and a good classification algorithm will maximize both precision and recall simultaneously. Thus, moderately good performance on

both will be favored over extremely good performance on one and poor performance on the other [21].

4.3.2 *METEOR Score:*

The **METEOR** (Metric for Evaluation of Translation with Explicit ORdering) score is a metric designed to evaluate the quality of machine-generated translations by comparing them to human reference translations. Unlike BLEU, METEOR addresses several limitations by incorporating precision, recall, and synonymy, along with a penalty for fragmentation. Precision measures the fraction of matched n-grams in the generated translation, while recall measures the fraction of matched n-grams in the reference translations. The metric calculates the harmonic mean of precision and recall, known as the F-score, where the parameter α balances these two aspects [3]. METEOR also applies a penalty for fragmented translations, which is based on the proportion of contiguous sequences, adjusting the final score to reflect the quality of the alignment.

The final METEOR score combines precision, recall, and fragmentation penalty to provide a comprehensive evaluation of translation quality. The formula for the METEOR score is given by

$$\text{METEOR Score} = F \cdot (1 - \text{Penalty}),$$

where the F-score is calculated as

$$F = \frac{(1 + \alpha^2) \cdot P \cdot R}{\alpha^2 \cdot P + R}.$$

This metric incorporates various linguistic features such as synonymy and stemming, enhancing its flexibility and accuracy in evaluating translations [3].

4.4 Results:

4.4.1 *Task-1:*

The validation performance of different models highlights notable variations in their effectiveness. The DeBERTa model achieved the highest validation accuracy and F1 score, with values of 0.745 and 0.7441, respectively. This indicates that DeBERTa is the most effective model for this task, showcasing its ability to accurately predict the target labels and align closely with the reference annotations.

RoBERTa also performed well, achieving a validation accuracy of 0.725 and an F1 score of 0.7241. While slightly lower than DeBERTa, RoBERTa's performance still demonstrates a strong capability in handling the nuances of the data, making it a competitive choice for this task.

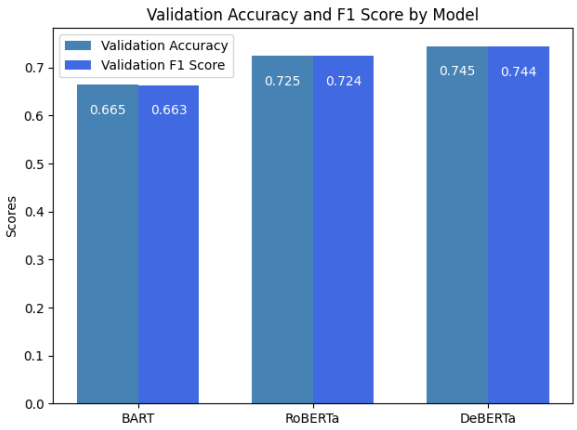

Figure 4: Accuracy and F-1 scores comparison of models.

Table 1: Weighted Average Metrics for Different Models

Model	Precision	Recall	F1 Score
BART	0.67	0.67	0.66
RoBERTa	0.73	0.72	0.72
DeBERTa	0.75	0.74	0.74

In contrast, the BART model, with a validation accuracy of 0.665 and an F1 score of 0.6635, exhibited comparatively lower performance. This suggests that BART, while still effective, may not capture the complexities of the data as well as DeBERTa and RoBERTa. The lower scores for BART could be attributed to several factors, such as suboptimal hyperparameters or overfitting issues.

Overall, these results underscore the superiority of the DeBERTa model for this specific task, followed closely by RoBERTa. The performance differences among the models highlight the importance of model selection and fine-tuning in achieving optimal results.

4.4.2 *Task-2:*

The METEOR scores for the different models on the validation data highlight notable variations in performance. The BART model with a learning rate of 5×10^{-5} achieved the highest METEOR score of 0.44. This result suggests that this learning rate configuration was most effective for gen-

erating spoilers that closely align with the reference texts, indicating a well-tuned model capable of better capturing the nuances in the data. In contrast, the BART model with a lower learning rate of 1×10^{-5} achieved a score of 0.39, reflecting a minor drop in performance, which may be attributed to less effective training or convergence issues.

The BART_large_cnn model underperformed with a METEOR score of 0.21. This lower score could be indicative of overfitting or suboptimal hyperparameter choices, which negatively impacted its ability to generate high-quality spoilers. On the other hand, the T5 base model scored 0.41, demonstrating a competitive performance but still falling short compared to the higher-performing BART model with a learning rate of 5×10^{-5}.

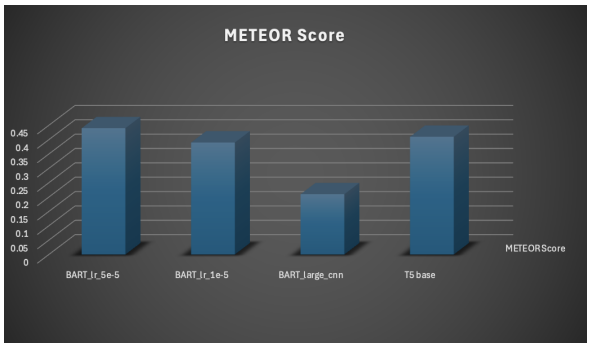

Figure 5: METEOR scores of different models on the validation data. BART with a learning rate of 5×10^{-5} achieved the highest score, followed by T5 base and BART with a learning rate of 1×10^{-5}. The BART_large_cnn model performed the worst.

Overall, these results underscore the effectiveness of the BART model with the higher learning rate for this task.

The project has been successfully submitted to the Kaggle leaderboard, where it received a good scores, further validating the chosen models' configuration and their practical applicability in generating effective clickbait spoilers and classifying them.

5 Conclusion

In this report, we explored various Transformer-based architectures for the NLP classification and generation tasks and compared their performances. Our findings indicated that Transformer-based models significantly outperform conventional neural network architectures. Among the models evaluated, DeBERTa achieved the highest performance with a validation F1 score of 0.7441 and a validation accuracy of 74.5% when it comes to classification, closely followed by RoBERTa and BART. For the generation task, the BART-base model performed the best among the models with a METEOR score of 0.44 followed by T5-base model with 0.41.

We used out of the box implementations of sequence classification and text generation of these models . For future work, we recommend exploring customized models that leverage BART, RoBERTa, DeBERTa and T5 encodings in combination with other handcrafted features to further enhance performance. Additionally, investigating the use of ELMo and ULMFiT embeddings for deep learning models could provide further improvements.

These findings underline the potential of advanced Transformer-based models in achieving high performance in NLP tasks, paving the way for more sophisticated approaches in future research.

The project is hosted on GitHub: https://github.com/praveen-rds/MSCI-641-Final-Project.

References

[1] Amol Agrawal. Clickbait detection using deep learning. In *2016 2nd international conference on next generation computing technologies (NGCT)*, pages 268–272. IEEE, 2016.

[2] Peter Bourgonje, Julian Moreno Schneider, and Georg Rehm. From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 EMNLP workshop: natural language processing meets journalism*, pages 84–89, 2017.

[3] Mateusz Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 376–380. Association for Computational Linguistics, 2014.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[5] Maik Frobe. Clickbait challenge at semeval. https://pan.webis.de/semeval23/pan23-web/clickbait-challenge.html, 2022.

[6] M. S. Z. R. Computer Science Graduate. What is bert: Bert for text classification, jun 2020. Accessed: 2024-08-06.

[7] Yuanliang He, Wei Wu, Yixin Zhang, et al. Deberta: Decoding-enhanced bert with disentangled attention. In *Proceedings of the 44th International Conference on Machine Learning*, 2021.

[8] R. Horev. Bert explained: State of the art language model for nlp. https://example.com/bert-explained, November 2018.

[9] Hugging Face. Hugging face – on a mission to solve nlp, one commit at a time, 2024. Accessed: 2024-08-06.

[10] P. Suleiman Khan. Bert, roberta, distilbert, xlnet - which one to use? https://example.com/bert-roberta-distilbert-xlnet, October 2019.

[11] Patrick Lewis, Ethan Perez, Alec Radford, et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, 2020.

[12] Yixin Liu, Myle Ott, Naman Goyal, et al. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[13] Yixin Liu, Myle Ott, Naman Goyal, et al. Multilingual denoising pre-training for neural machine translation. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, 2020.

[14] Yixin Liu, Myle Ott, Naman Goyal, et al. Roberta: A robustly optimized bert pretraining approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, 2020.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[16] Colin Raffel, Noam Shazeer, Adam Roberts, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. In *Journal of Machine Learning Research*, volume 21, pages 1–67, 2020.

[17] Gaurav Sahu. [task 1] clickbait detection msci641, s24, 2024.

[18] Gaurav Sahu. [task 2] clickbait detection msci641, s24, 2024.

[19] Timo Schick and Hinrich Schütze. Exploiting cloze questions for few-shot text classification. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, 2020.

[20] Kai Shu, Suhang Wang, Thai Le, Dongwon Lee, and Huan Liu. Deep headline generation for clickbait detection. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 467–476. IEEE, 2018.

[21] C.J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[23] A. Wang, A. Singh, J. Michael, et al. Roberta vs. bert: A comprehensive comparison for language understanding tasks. *Journal of Machine Learning Research*, 21:1–32, 2020.

[24] Mateusz Woźny and Mateusz Lango. Generating clickbait spoilers with an ensemble of large language models. In *Proceedings of the 16th International Natural Language Generation Conference*. Association for Computational Linguistics, 2023.

[25] Hengrong Wu, Min Zhang, Yi Yang, et al. Bart for text generation: How to fine-tune on domain-specific data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP 2021)*, 2021.